Amendment to the Claims

1. (currently amended) A method, comprising:

(a) maintaining on a network interface device a DMA command queue;

(b) a processor on the network interface device causing a value to be pushed onto the DMA command queue, the value being indicative of corresponding DMA command;

(c) popping a value off the DMA command queue, a DMA controller on the network interface device then executing a DMA command indicated by the popped value;

(d) repeating (b) and (c) such that a first portion of data is transferred from host storage to a local memory on the network interface device and such that a second portion of data is transferred from the host storage to the local memory on the network interface device; ~~and~~

(e) maintaining a DMA command complete queue on the network interface device, the DMA controller pushing values onto the DMA command complete queue, the processor popping values off the DMA command complete queue; and

(f) after both the first portion and the second portion are present in the local memory, outputting the first and second portions of data from the network interface device to a network, the first and second portions making up at least a part of a data payload of a network communication.

2. (canceled)

3. (currently amended) The method of Claim 1[[2]], wherein the processor uses the DMA command complete queue to determine that the first and the second portions of data are both present in local memory on the network interface device.

4. (currently amended) The method of Claim 1, wherein the network interface device comprises queue manager hardware, the queue manager hardware maintaining the DMA command queue and the DMA command complete queue in static random access memory (SRAM), wherein the processor, the queue manager hardware, the SRAM, and the DMA controller are all part of the same integrated circuit.

5. (previously presented) A method, comprising:

(a) maintaining on a network interface device a DMA command queue;

(b) a processor on the network interface device causing a value to be pushed onto the DMA command queue, the value being indicative of corresponding DMA command;

(c) popping a value off the DMA command queue, a DMA controller on the network interface device then executing a DMA command indicated by the popped value;

(d) repeating (b) and (c) such that a first portion of data is transferred from host storage to a local memory on the network interface device and such that a second portion of data is transferred from the host storage to the local memory on the network interface device; and

(e) after both the first portion and the second portion are present in the local memory, outputting the first and second portions of data from the network interface device to a network, the first and second portions making up at least a part of a data payload of a network communication, wherein the pushing of (b) occurs twice before the popping of (c) occurs once.

6. (original) The method of Claim 1, wherein each of the values on the DMA command queue is a DMA command.

7. (original) The method of Claim 1, wherein each of the values on the DMA command queue comprises a pointer to a DMA command.

8. (original) The method of Claim 1, wherein each of the values on the DMA command queue is a number that identifies a location where a DMA command is stored.

9. (original) The method of Claim 1, wherein the network interface device is an expansion card coupled to a host computer, the host storage being part of the host computer.

10. (original) The method of Claim 1, wherein the network interface device is a part of a host computer.

11. (currently amended) A network interface device, comprising:

queue manager hardware that maintains a DMA command queue and a DMA command complete queue;

a processor coupled to the queue manager hardware, the processor causing values to be pushed onto the DMA command queue, the processor causing values to be popped from the DMA command complete queue;

a DMA controller coupled to the queue manager hardware, the DMA controller executing DMA commands, the DMA commands executed being indicated by values popped off the DMA command queue, the DMA controller pushing values onto the DMA command complete queue;

local memory that temporarily stores a first portion of data transferred by execution of one or more DMA commands from data storage on a host coupled to the network interface device into the local memory, the local memory also temporarily storing a second portion of data transferred by execution of one or more DMA commands from the data storage on the host to the local memory; and

physical layer interface and media access control circuitry, the physical layer interface and media access control circuitry outputting the first and second portions of data from the network interface device to a network, the first and second portions of data being output in the form of a data payload of a network communication.

12. (original) The network interface device of Claim 11, wherein the local memory is dynamic random access memory (DRAM).

13. (currently amended) The network interface device of Claim 12, wherein the queue manager hardware stores at least part of the DMA command queue and the DMA command complete queue in static random access memory (SRAM).

14. (currently amended) A method, comprising:

(a) maintaining on a network interface device a DMA command queue, the DMA command queue being maintained by queue manager hardware on the network interface device;

(b) a processor on the network interface device causing a value to be pushed onto the DMA command queue, the value being indicative of corresponding DMA command;

(c) popping a value off the DMA command queue, a DMA controller on the network interface device then executing a DMA command indicated by the popped value;

(d) repeating (b) and (c) such that a first portion of data is transferred from a first place on the network interface device to a second place on the network interface device, and such that a second portion of data is transferred from the first place on the network interface device to the second place on the network interface device;

(e) maintaining a DMA command complete queue on the network interface device, the DMA controller pushing values onto the DMA command complete queue, the processor popping values off the DMA command complete queue;

(f) after both the first portion and the second portion have been transferred to the second place in (d), the processor taking a software branch; and

(g[[f]]) after taking the software branch, the processor outputting the first and second portions of data from the network interface device.

15. (original) The method of Claim 14, wherein the processor outputs the first and second portions of data in (f) to a network.

16. (original) The method of Claim 14, wherein the processor outputs the first and second portions of data in (f) to a host computer.

17. (original) The method of Claim 14, wherein the first place is a dynamic random access memory (DRAM) and wherein the second place is a bus interface.

18. (original) The method of Claim 14, wherein the first place is a bus interface and wherein the second place is a dynamic random access memory (DRAM).

19. (original) The method of Claim 14, wherein the first place is a bus interface and wherein the second place is a static random access memory (SRAM).

20. (original) The method of Claim 14, wherein the first place is a static random access memory (SRAM) and wherein the second place is a bus interface.

21. (currently amended) A method, comprising:

(a) using a DMA command queue to ensure that a plurality of DMA moves are completed in a particular sequence, each of the DMA moves being a move of information from one location on a network interface device to another location on the network interface device, the DMA command queue being maintained by queue manager hardware on the network interface device, the DMA moves being carried out by a DMA controller, the DMA controller being a part of the network interface device; ~~and~~

(b) the DMA controller pushing values onto a DMA command complete queue, each value indicating that one of the DMA moves has been carried out by the DMA controller;

(c) a processor on the network interface device popping values off the DMA command complete queue; and

(d) outputting at least part of the information from the network interface device.

22. (previously presented) The method of Claim 21, wherein the information output in (b) is output from the network interface device to a host computer, the host computer being coupled to the network interace device.

23. (original) The method of Claim 21, wherein the information output in (b) is output from the network interface device to a network.

24. (original) The method of Claim 21, wherein a first of the plurality of DMA moves is a move of at least a part of a frame of a session layer message, and wherein a second of the plurality of DMA moves is a move of at least a part of a subsequent frame of the session layer message.

25. (previously presented) A method, comprising:

(a) using a DMA command queue to ensure that a plurality of DMA moves are completed in a particular sequence, each of the DMA moves being a move of information from one location on a network interface device to another location on the network interface device, the DMA command queue being maintained by queue manager hardware on the network interface device, the DMA moves being carried out by a DMA controller, the DMA controller being a part of the network interface device; and

(b) outputting at least part of the information from the network interface device, wherein a first of the plurality of DMA moves is a move of at least a part of a frame of a session layer message, and wherein a second of the plurality of DMA moves is a move of at least a part of a subsequent frame of the session layer message, wherein a processor pushes values onto the DMA command queue, each of the values being indicative of a different DMA command, and wherein the processor analyzes at least a part of the information moved in the first move, the processor doing the analyzing after the first move is complete.

26. (Original) The method of Claim 25, wherein said one location on the network interface device is a dynamic random access memory (DRAM) and wherein said another location on the network interface device is a static random access memory (SRAM), wherein the first move is a move of information from a first buffer in the DRAM to a first buffer in the SRAM, and wherein the second move is a move of information from a second buffer in the DRAM to a second buffer in the SRAM.

27. (Original) The method of Claim 25, wherein the processor does the analyzing before the second move is complete.

28. (currently amended) An apparatus, comprising:

means for maintaining a DMA command queue and a DMA command complete queue on a network interface device and for causing values to be pushed onto the DMA command queue and for causing values to be popped from the DMA command complete queue, each of the values pushed onto the DMA command queue being indicative of one of a plurality of DMA commands; and

a DMA controller that executes the plurality of DMA commands such that the DMA commands are completed in a particular order, the DMA controller pushing values onto the DMA command complete queue, each of the values pushed onto the DMA command complete queue indicating completion of one of the plurality of DMA commands.

29. (original) The apparatus of Claim 28, wherein the means comprises a processor.

30. (original) The apparatus of Claim 29, wherein the means further comprises a hardware queue manager.

31. (currently amended) A method, comprising:

(a) pushing values onto a DMA command queue in an order, each of the values being indicative of a different one of a plurality of DMA commands, wherein the DMA command queue is maintained on a network interface device (NID), and wherein the NID performs fast-path transport and network layer protocol processing; ~~and~~

(b) popping the DMA command queue such that a DMA controller on the NID executes the plurality of DMA commands in the order in which the associated values were pushed onto the DMA command queue, the DMA controller being a part of the NID;[[.]]

(c) the DMA controller pushing values onto a DMA command complete queue, each of the values pushed onto the DMA command complete queue indicating completion of one of the plurality of DMA commands; and

(d) popping the DMA command complete queue.

32. (currently amended) The method of Claim 31, wherein the NID includes a first memory and a second memory, the method further comprising:

(e[[c]]) receiving multiple frames of a session layer network message onto the NID and storing the frames in the first memory, wherein execution of one of the plurality of DMA commands in (b) results in a move of at least a part of one of the frames from the first memory to the second memory, and wherein execution of another of the plurality of DMA commands in (b) results in a move of at least a part of another of the frames from the first memory to the second memory.

33. (previously added) The method of Claim 31, wherein the NID is integrated into an integrated circuit taken from the group consisting of: a memory controller integrated circuit, a graphics controller integrated circuit, an input/output integrated circuit, and a bridge integrated circuit, and wherein the integrated circuit of which the NID is a part is realized on a motherboard of a host computer.

34. (currently amended) The method of Claim 32[[31]], wherein the DMA command complete queue is used to determine that the at least a part of one of the frames and the at least a part of another of the frames are both present in the second memory ~~wherein the NID is a means for performing fast-path transport and network layer protocol processing~~.